模型驱动下的 Web 应用系统自动生成

王海林[1,2]
(1. 山西财经大学 信息管理学院,山西 太原 030006;
2. 奥塔哥大学 信息科学系,新西兰,但尼丁 奥塔哥 9054)

摘 要:为了提高 Web 应用系统开发效率,提出了模型驱动下的 Web 应用系统自动生成方法。这种生成方法是以 MetaEdit+作为元建模工具,首先创建 Web 应用系统元模型、定制 DSL,进而建立 Web 应用系统领域模型,然后通过 MetaEdit+提供的生成器定义语言 MERL,软件开发人员可以很方便地设计出 Web 应用系统所需的 JSP 生成器、Servlet 生成器、Javabeans 生成器和数据库生成器,从 Web 应用系统图形模型直接生成整个 Web 应用系统。最后通过一个 WebShopping 实例详细介绍了模型驱动下的 Web 应用系统生成方法及生成过程。经测试,所生成的 Web 应用系统可以在 Windows 操作系统中的 Web 应用服务器上正确运行。

关键词:模型驱动;特定领域建模语言;Web 应用系统;元建模;MetaEdit+
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2012)01-0083-04

Model Driven Automatic Generation of Web Application Systems

WANG Hai-lin[1,2]
(1. College of Information Management, Shanxi University of Finance and Economics, Taiyuan 030006, China;
2. Department of Information Science, University of Otago, Dunedin 9054, New Zealand)

Abstract: In order to promote software development efficiency, it proposes an approach of model driven automatic generation of Web applications. The approach takes MetaEdit+ as a meta-modeling tool. The first step the approach suggests is to build Web application meta-models and to customize DSL. The further step is to build Web application domain models by DSL. Then by using generator definition language MERL which MetaEdit+ provides, the software developers can design conveniently JSP generator, Servlet generator, Javabeans generator and database generator that Web application systems need. These generators can produce the whole Web application system directly from the Web application graph models. Finally, the approach of model driven automatic generation of Web applications will be introduced in detail through an instance named WebShopping. The test result indicates that the generated Web application can run correctly on the Web application server in the Windows operating system environment.

Key words: model driven; domain specific language; Web application system; Meta-Modeling; MetaEdit+

0 引 言

程序设计自动化一直以来是软件设计人员梦寐以求的事情,也是软件工程追求的目标之一。程序设计自动化的高级阶段是自动生成整个软件系统,文中提出的模型驱动下的 Web 应用系统自动生成就是在这方面所做的初步尝试。模型驱动下的 Web 应用系统自动生成是一种从领域模型(图形模型)自动生成整个 Web 应用系统的方法,它的基本思想源于模型驱动开发方法 MDD(Model Driven Development)[1]。MDD 与基于统一建模语言 UML(Unified Modeling Language)的方法都是以模型为核心的系统开发方法,但二者有很大的不同,MDD 更注重应用领域,针对不同应用领域的特点建立不同模型,所以 MDD 往往更为有效。具体地讲,就是根据不同应用领域的特点,首先设计出最适合于该领域的建模语言,称其为 DSL(Domain Specific Language)[2],然后再对领域建模,这样可以更快速有效地解决实际问题,更容易满足用户的需求。自行设计 DSL 的好处是可以让模型"活"起来,可以从模型中获得自动生成应用系统所需的各种数据,而这些 UML 是无法做到的。因此,模型驱动下的 Web 应用系统自动生成方法需要先设计 DSL,而设计 DSL 又要先建立元模型(meta-model),即描述模型的模型。通常使用元建模工具(meta-modeling tools)来设计元模型,目前元建模工具有很多,比较著名的有 MetaEdit+、GME、GMF、EMF、DOME 等[3],从易用性和代码生

收稿日期:2011-06-09;**修回日期**:2011-09-15
基金项目:山西省省筹公派出国留学资金(晋留管办发 2009-4 号)
作者简介:王海林(1962-),男,山西大同人,副教授,硕士,研究方向为模型驱动软件开发、元建模、领域本体建模、XML 数据库等。

成等因素考虑,笔者选择了 MetaEdit+。

1 元建模工具 MetaEdit+

MetaEdit+是一款使用较广的商用元建模工具,第一个商用版本 MetaEdit+1.1 发布于 1993 年,目前最新版本为 MetaEdit+4.5。和 UML 一样,MetaEdit+是一个基于图形的建模工具,它提供了一组建模工具,如 diagram editors, table editor, matrix editor, browsers, report and code generation, method tools, object repository, API & XML connectivity 等[4]。使用 MetaEdit+可以大大缩短软件开发周期,软件开发效率提高 5-10 倍[5],因为它提供了强大代码生成功能。通过 MetaEdit+实现 Web 应用系统的自动生成可分为以下步骤:

1)对拟生成的应用系统进行需要分析,确定用于系统的系统结构、设计模式等;

2)建立元模型,定制 DSL(MetaEdit+采用定制方法[3]生成 DSL);

3)使用定制好的 DSL 建立领域模型;

4)设计应用系统生成器;

5)运行系统生成器生成应用系统。

2 拟生成的 Web 应用系统

以一个网上商品销售系统 WebShopping 作为拟生成的目标系统。考虑到系统生成方法的通用性和所生成目标系统的实用性,目标系统 WebShopping 的体系结构、设计模式和数据访问技术都是目前较为流行的,下面就详细介绍一下这几个方面。

2.1 系统体系结构

WebShopping 采用了 Browser/Web/Database 三层体系结构,如图 1 所示。GlassFish V3 作为 Web 应用程序服务器,它是一款开源的、商业兼容的、以达到产品级质量[6]的 Web2.0 开发和部署平台,它的性能和受用户欢迎程度都优于 Tomcat 6。数据库使用了新近推出的 H2 开源数据库[7],它对 SQL 的支持优于 MySQL。H2 是采用 Java 编写的,有良好的跨平台性,且支持几乎所有主要的数据库特性。

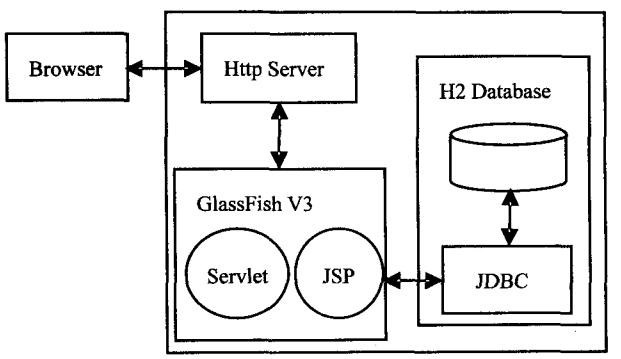

图 1 WebShopping 系统结构图

2.2 系统设计模式

WebShopping 系统采用了目前 B/S 结构的 Web 应用系统在中间层上常采用的 MVC(Model-View-Controller[8],即模型视图控制器)设计模式。Model(即 JavaBeans)层用来实现业务逻辑、访问数据库,View(即 JSP)层用来显示用户界面、接受和响应用户请求,Controller 层负责 View 层和 Model 层之间的控制关系。

2.3 数据访问技术

在 WebShopping 系统中连接数据库采用了 Jdbc 数据库连接池技术(Jdbc Connection pooling)[9],这样,当用户较多时可以让他们共享较少的数据库连接,从而大大提高数据库的访问速度和使用效率。

3 基于 MetaEdit+的元建模

在设计 Web 应用系统代码生成器之前,首先要建立领域模型。在 UML 中通常用类图对静态结构建模,用顺序图,状态图对动态行为建模[10],而 MetaEdit+是一种元建模工具,没有事先将图分为类图、顺序图或状态图等,图的含义是由建模者自己定义。通过 MetaEdit+提供的工具设计自己风格的图形符号和图。

文中建立领域元模型就是设计绘制数据库概念模型(即 Entity-Relationship 图[11])、面向对象模型(如类图[12])和代码生成所需的各种元素及其图形符号。具体地讲,就是创建 MetaEdit+ 的 Property(属性)、Object(对象)、Relationship(联系)、Role(角色)和 Graph(图)等,这里只能以 Graph(图)的创建来说明。

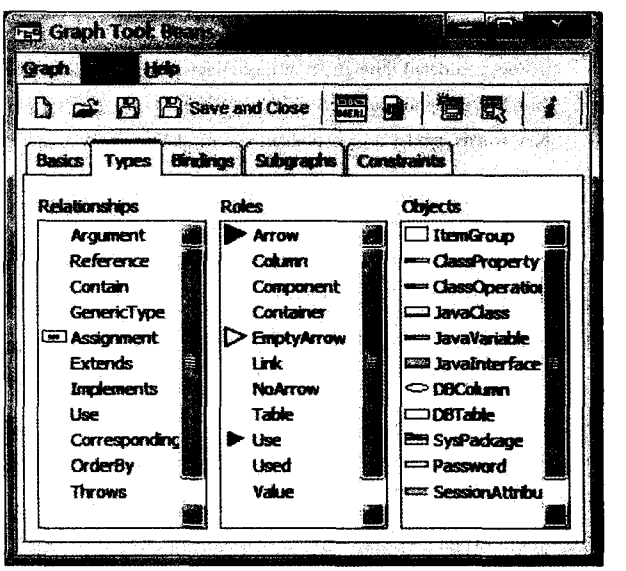

图 2 Graph Tool 的设计界面

绘制各种图形之前首先要建立图形元模型,即 Graph。MetaEdit+提供了 Graph Tool 工具来建立图形元模型。图 2 给出了 Graph Tool 的设计界面通过 Graph Tool 可以定义创建图所需的各种对象、联系、角色以及它们之间的绑定关系,还可以定义图形包含的

子图形和各种约束条件。

4　建立领域模型

元模型建好后,就可以通过 Diagram Editor(Diagram Editor 相当于定制好的 DSL,针对不同类型的元模型 Graph 它会给出不同的设计界面)工具来建立领域模型了。考虑到目标系统采用 MVC 设计模式,所以这里领域模型设计分为四部分。

1. Servlet 部分。

定义了一组 Servlet 对象,包括:

AddToOrderServlet、BuyServlet、ConfirmOrderServlet、CreateAccountServlet、LoginServlet、LogoutServlet

2. JSP 部分。

定义了各种类型的对象:

XML 类:Web

HTML 类:Login、CreateAccount

JSP 类:Error、Thanks、NotEnoughStock、QuantityToBuy、index、ViewProducts、Checkout

JSPF 类:AuthenticationCheck、NavigationMenu

CSS 类:Forms、style

Properties 类:Bundle

3. JavaBeans。

定义了四个包,每个包包括若干类或接口。

domain 包:包括 Customer、Product、Order、OrderItem 类,分别表示顾客、商品、订购和订购商品条目。图 3 给出了 Order 类的属性和方法图,其中 items、datetime、customer 为类属性,均为 private("-"号代表 private,"+"号表示 public,"#"号表示 protected)。datetime 的数据类型为 Timestamp,用来存储商品的订购日期和时间(orderdatetime 是数据库概念模型中的属性,datetime 与 orderdatetime 之间的联系是 Corresponding,它可以定义类属性与实体集属性之间的对应关系,这种对应关系在关系模型与面向对象模型进行数据传递和转换时非常有用);customer 的数据类型为 Customer,用来表示订购商品的顾客;items 的数据类型为泛型 Collection <OrderItem>,并初始化为 new HashSet<OrderItem>(),用来存放订购的商品。getTotal、addItem、Order 为类方法,均为 public。getTotal 用以计算订购商品的总价,返回 Double 数据类型;addItem 是将订购商品添加到列表中,无返回值;Order 为补充的构造方法。该应用系统生成器可以生成类的构造方法来初始化类对象(同时也生成了每个类属性的 get 和 set 方法,因为这些方法都是自动生成的,所以图 4 中就没有给出),如果一个类需要多个构造方法,就要在图中补充定义了。customer 和 item 分别是方法 Order 和 addItem 的形参 java. sql. Timestamp、java. util. Date、java. util. Collection 和 java. util. HashSet 分别是 Java 系统包中的类和接口,生成 Java 代码时生成器会自动搜索到这些类和接口并用 import 语句将它们导入到文件。与类属性不同,每个类方法都连接着一个子图(在建模型时,可用 MetaEdit+提供的 decompositions 或 explosions 将方法和子图连接,并在生成器中用 do decompositions 或 do explosions 命令从方法导航到子图),用来刻画方法的处理逻辑。

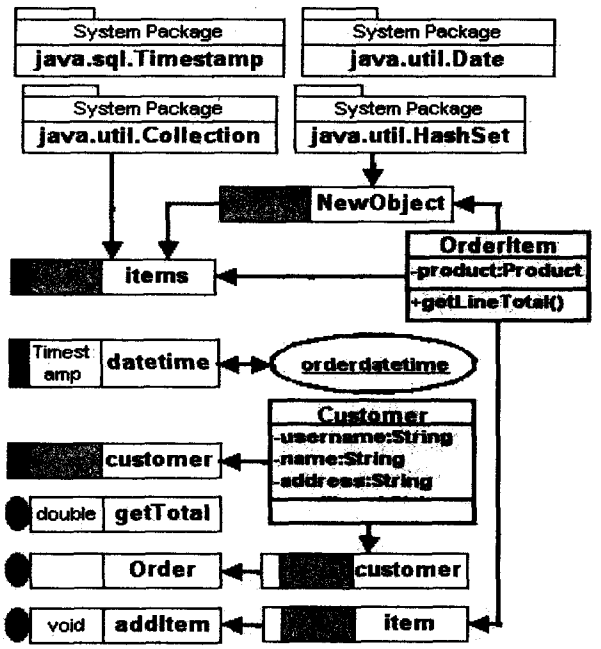

图 3　Order 类的属性和方法图

dao 包:包括 CustomerDAO、ProductDAO、OrderDAO、DAOException。DAO 是数据访问对象(Data Access Object)的缩写,CustomerDAO、ProductDAO 和 OrderDAO 是用来访问数据库的接口,而 DAOException 是数据访问异常处理类。

dao. jdbc 包:包括 JdbcConnection、AbstactJdbcDAO、CustomerJdbcDAO、ProductJdbcDAO、OrderJdbcDAO。

Servlets 包:包含了前面介绍的一组 Servlet 对象的实现类。

4. 数据库概念模型部分。

WebShopping 系统包括顾客 Customers、商品 Products 和订购 Orders 实体集,各实体集的属性及其说明如表 1 所示。

表 1　各实体集属性说明

Customers	属性说明	Products	属性说明	Orders	属性说明
customerid	顾客编号	Productid	商品号	customerid	顾客编号
name	顾客姓名	Name	品名	productid	商品号
address	住址	Description	说明	orderdatetime	订购日期时间
creditcard	信用卡号	Category	分类	quantity	订购数量
password	密码	Price	价格		
		Quantity	库存数量		

5 应用系统生成器设计

模型建好后，接下来就是设计生成器从模型直接生成 Web 应用系统所需的 Java 程序代码和各种其他类型的文件（如. xml、. html、. jsp、. jsf、. css 等），编译 Java 程序代码后就成为可运行的 Web 应用系统了。

MetaEdit+提供了强大的代码生成功能，包括生成器定义语言 MERL 和生成器设计器 Generator Editor。图 4 给出了 WebShopping 系统生成器的层次结构。

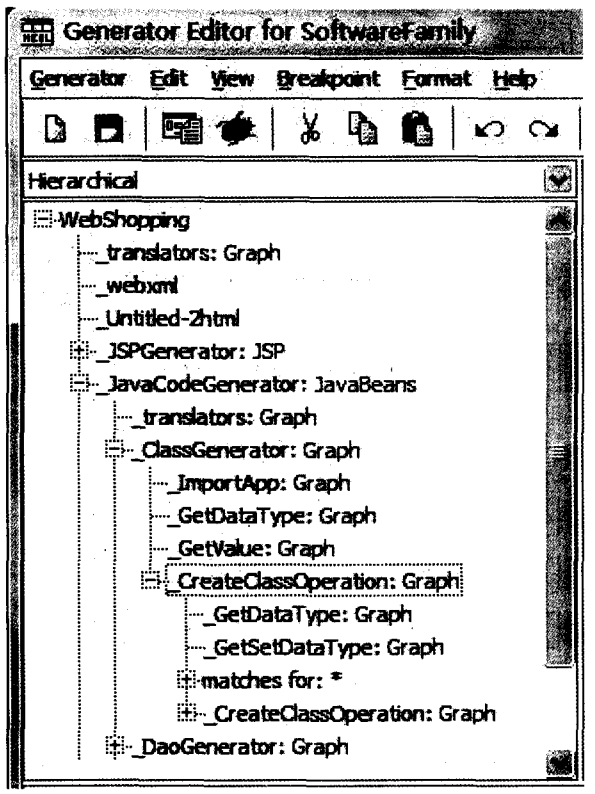

图 4 WebShopping 系统生成器的结构图

系统生成器主要分为四部分，即四个子生成器，其中子生成器_JSPGenerator 主要负责生成. jsp 类型文件，除了. jsp 类型文件，它还可以生成. jspf、. html、. css 等类型文件；子生成器_JavaCodeGenerator 负责生成 Java 源代码文件（生成器也可以将源代码文件编译成字节码文件，但建议使用目前流行的 *Netbeans6. 5* 来编译，它的编译器设计得更加完善）；子生成器_webxml 用来生成 web. xml 文件，该文件是负责注册和映射每个 Servlet 的；子生成器_DatabaseGenerator 是用来生成数据库的（准确地说，是用来生成数据表的，因为用 E-R 图描述的数据库概念模型并没有给出数据库名，另外与数据库建立连接也必须事先建好一个数据库，连接好数据库才能操作数据库，例如根据 E-R 图自动创建数据表），_DatabaseGenerator 是根据 E-R 图生成含有标准 SQL 语句的 Java 源代码文件，然后运行该文件，执行 SQL 语句达到创建数据表的目的，在建表的同时，也定义了表内的完整性约束（如实体完整性约束、用户定义完整性约束）和表间的完整性约束（参照完整性约束）。

6 结束语

文中所述内容是笔者在新西兰奥塔哥大学进修期间的研究成果，经过数个月的努力，笔者设计出了一套较为完整的 Web 应用系统元模型和系统生成器。所建的元模型和生成器，完全可以用于软件开发或建模类课程的教学中，也可以用于实际的 Web 应用系统开发。当然由于实际应用的复杂性，在使用的过程中肯定还会遇到各种各样的问题，需要不断地对元模型和生成器进行改进和完善。另外，在使用 MetaEdit+的过程中，笔者也发现了它的许多不足，并通过 Email 与 MetaEdit+的设计人员进行过多次交流，有些建议被采纳，承诺在未来的版本中进行改进。尽管模型驱动下的应用系统自动生成要做的事情还有很多，但它必将成为软件工程未来的一个发展方向，不仅可以提高软件开发速度，并使软件维护变得更简单更容易。

参考文献：

[1] soley R，Frankel D S，Parodi J. The MDA Journal：Model Driven Architecture Straight from the Masters[M]. New York：Meghan Kiffer Press，2004：135-146.

[2] Fowler M. Domain Specific Languages[M]. Toronto：Addison-Wesley Professional，2010.

[3] 刘 辉，麻志毅，邵维忠. 元建模技术研究进展[J]. 软件学报，2008，19(6)：1317-1327.

[4] 周金根. MetaModelEngine：元模型引擎开发思路[EB/OL]. [2010-10-12]. http://www. cnblogs. com/zhoujg/archive/2010/07/28/1786155. html.

[5] MetaCase. MetaEdit+ Domain-Specific Modeling Environment [EB/OL]. [2010-09-10]. http://www. metacase. com/MetaEdit. html.

[6] Venkat. GlassFish Overview[EB/OL]. [2011-01-25]. http://middlewareforum. com/weblogic/? p=861.

[7] Mueller T. H2 Database Engine[EB/OL]. [2010-10-20]. http://www. h2database. com/html/main. html.

[8] Reenskaug T. Working with Objects：The OOram Software Engineering Method[M]. New Jersey：Prentice Hall，1995.

[9] 周彩兰，孙 琳，李素芬. 基于 JSP 的网络数据库连接技术[J]. 计算机技术与发展，2006，16(4)：209-211.

[10] 李思广，林子禹，胡 峰，等. 基于 UML 的软件过程建模方法研究[J]. 计算机工程与应用，2003，39(6)：76-78.

[11] 王 珊，萨师煊. 数据库系统概论[M]. 第 4 版. 北京：高等教育出版社，2006.

[12] 牛丽平，郭新志，宋 强，等. UML 面向对象设计与分析基础教程[M]. 北京：清华大学出版社，2007.